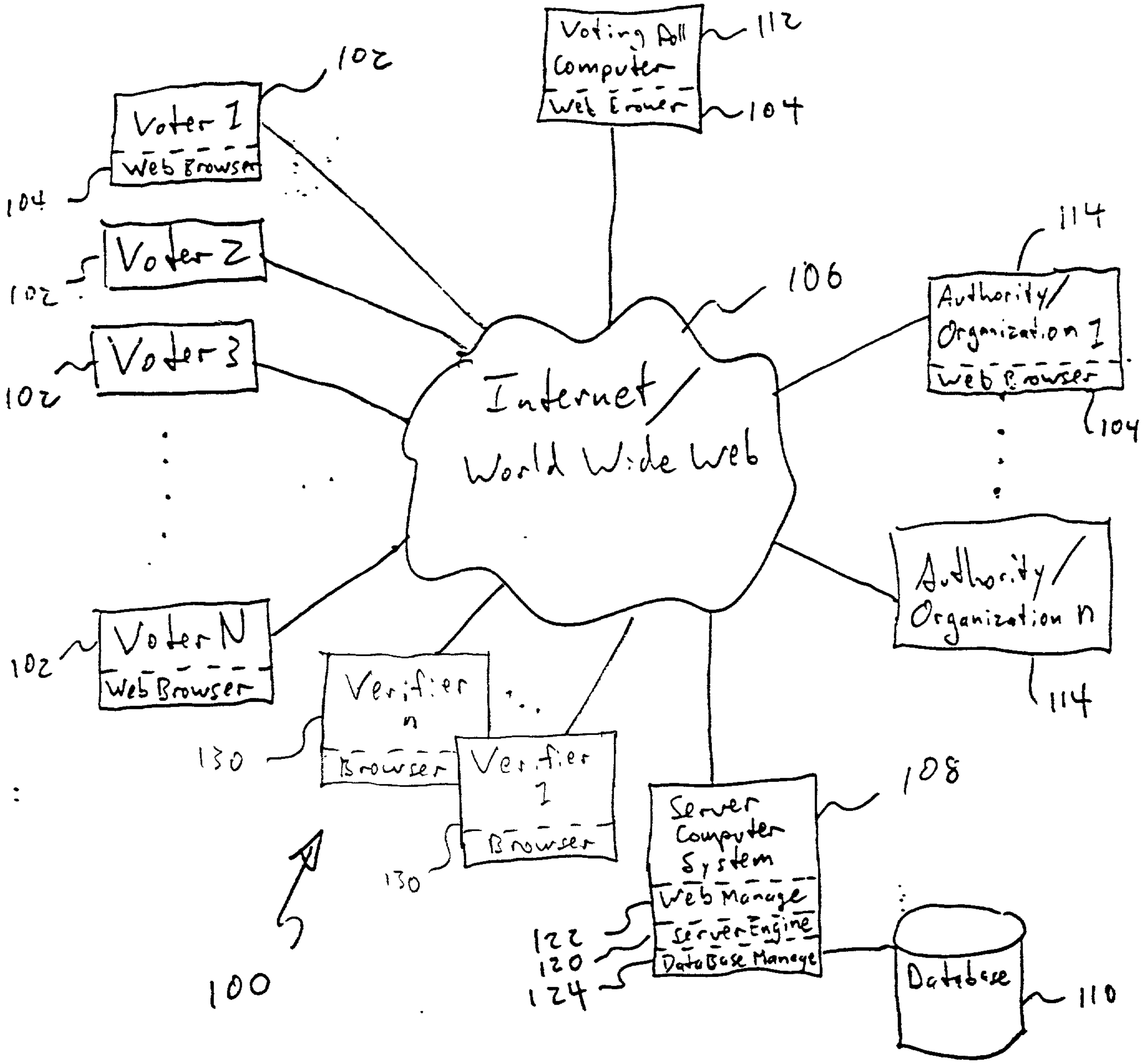
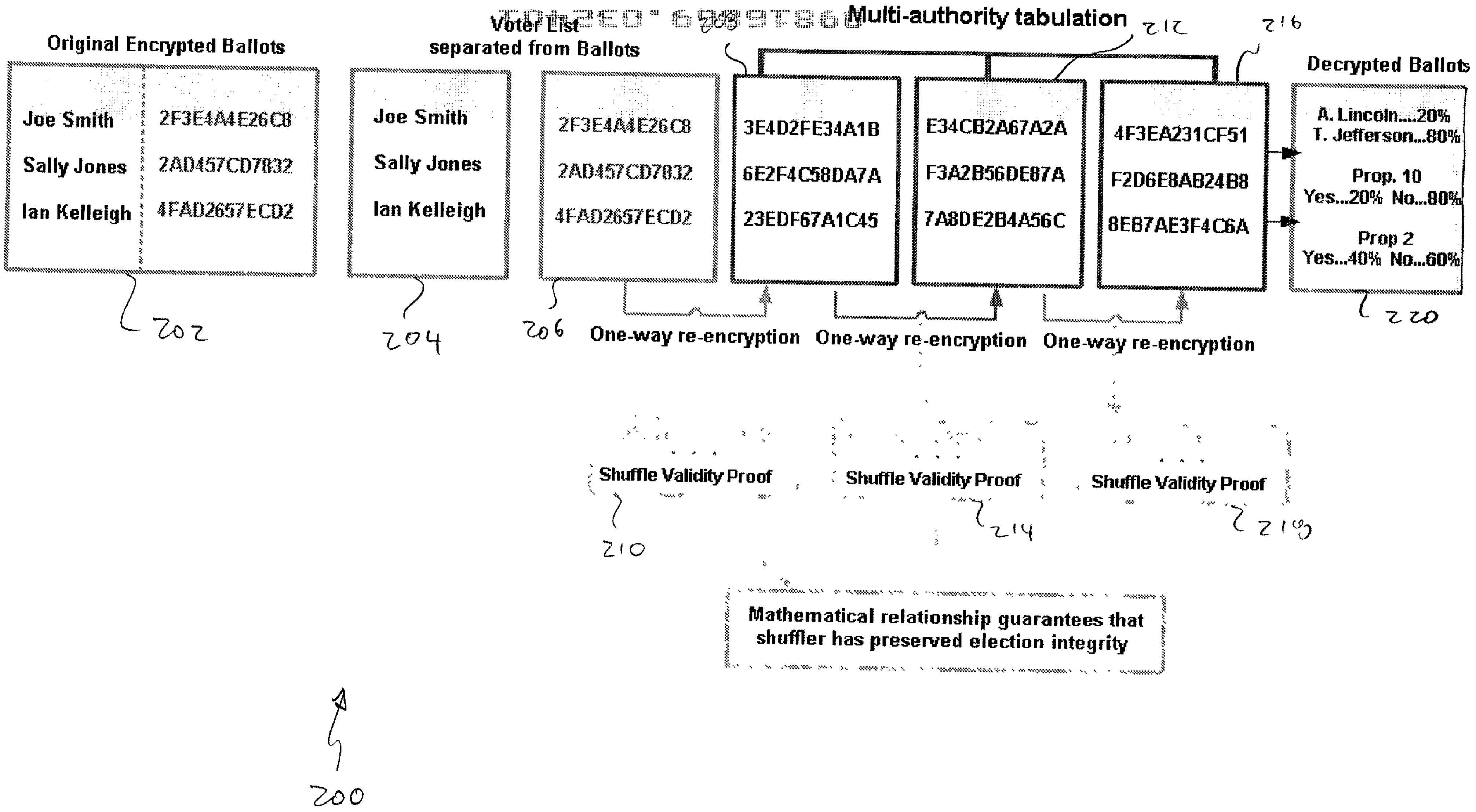

Voting Poll Computer
Web Brower
112
104
Voter 1
Web Browser
102
104
Voter 2
102
Voter 3
102
Voter N
Web Browser
102
Internet/ World Wide Web
106
Authority/ Organization 1
Web Browser
114
104
Authority/ Organization n
114
Verifier n
Browser
130
Verifier 1
Browser
130
Server Computer System
Web Manage
Server Engine
DataBase Manage
108
122
120
124
Database
110
100

FIG. 1



Original Encrypted Ballots
Joe Smith
Sally Jones
Ian Kelleigh
2F3E4A4E26C8
2AD457CD7832
4FAD2657ECD2
202
Voter List separated from Ballots
Joe Smith
Sally Jones
Ian Kelleigh
204
2F3E4A4E26C8
2AD457CD7832
4FAD2657ECD2
206
Multi-authority tabulation
208
212
216
3E4D2FE34A1B
6E2F4C58DA7A
23EDF67A1C45
E34CB2A67A2A
F3A2B56DE87A
7A8DE2B4A56C
4F3EA231CF51
F2D6E8AB24B8
8EB7AE3F4C6A
Decrypted Ballots
A. Lincoln....20%
T. Jefferson...80%
Prop. 10
Yes...20% No...80%
Prop 2
Yes...40% No...60%
220
One-way re-encryption
One-way re-encryption
One-way re-encryption
Shuffle Validity Proof
210
Shuffle Validity Proof
214
Shuffle Validity Proof
218
Mathematical relationship guarantees that shuffler has preserved election integrity
200

FIG 2

Scaled Iterated Logarithmic Multiplication Proof

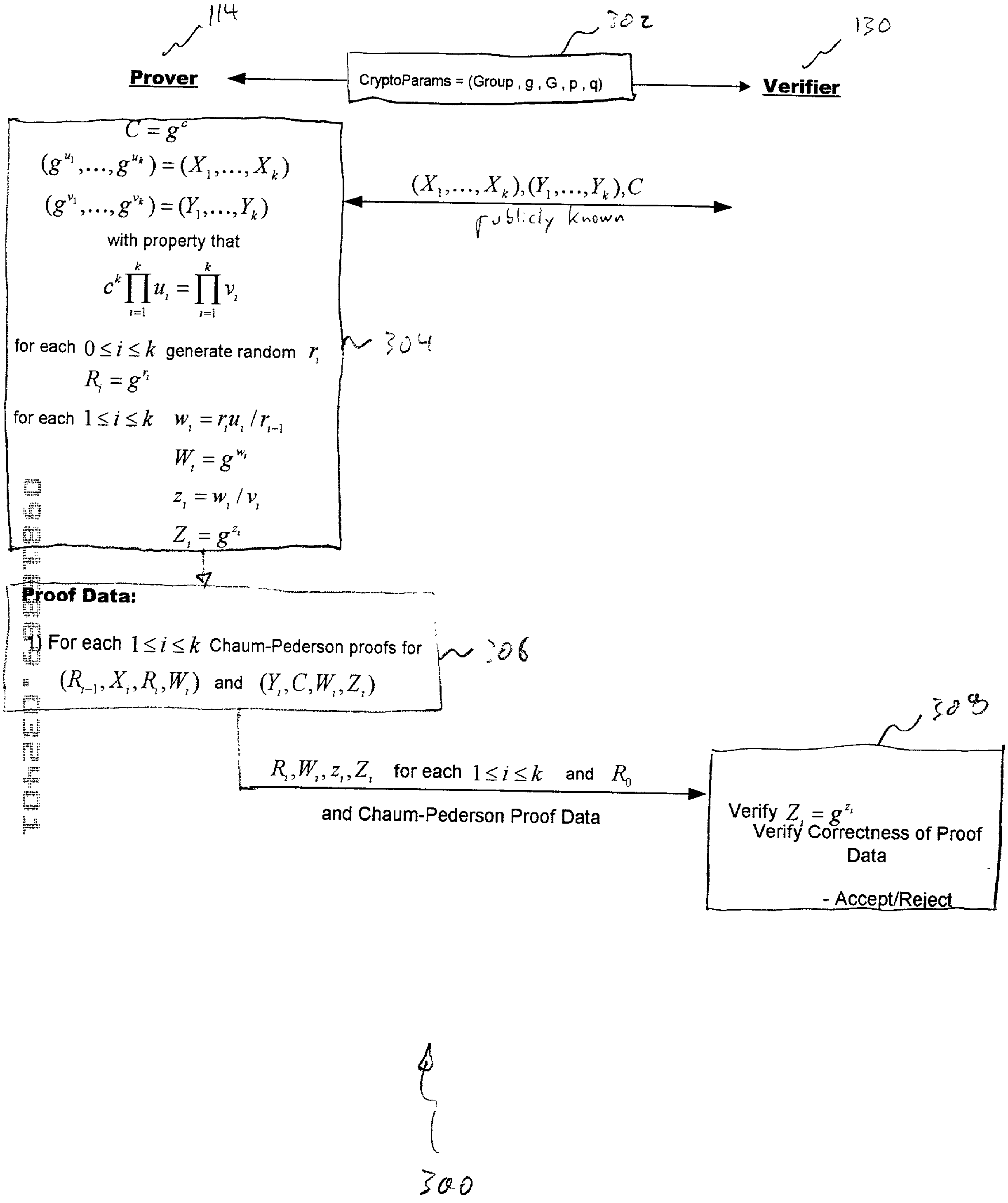

114
302
130
Prover
CryptoParams = (Group , g , G , p , q)
Verifier
$C = g^c$
$(g^{u_1},\dots,g^{u_k}) = (X_1,\dots,X_k)$
$(g^{v_1},\dots,g^{v_k}) = (Y_1,\dots,Y_k)$
with property that
$c^k \prod_{i=1}^{k} u_i = \prod_{i=1}^{k} v_i$
for each $0 \le i \le k$ generate random r_i
$R_i = g^{r_i}$
for each $1 \le i \le k$ $w_i = r_i u_i / r_{i-1}$
$W_i = g^{w_i}$
$z_i = w_i / v_i$
$Z_i = g^{z_i}$
$(X_1,\dots,X_k),(Y_1,\dots,Y_k),C$
publicly known
304
Proof Data:
1) For each $1 \le i \le k$ Chaum-Pederson proofs for
(R_{i-1},X_i,R_i,W_i) and (Y_i,C,W_i,Z_i)
306
308
R_i,W_i,z_i,Z_i for each $1 \le i \le k$ and R_0
and Chaum-Pederson Proof Data
Verify $Z_i = g^{z_i}$
Verify Correctness of Proof
Data
- Accept/Reject
300

FIG 3

Simple Shuffle
(Shuffler knows exponents)

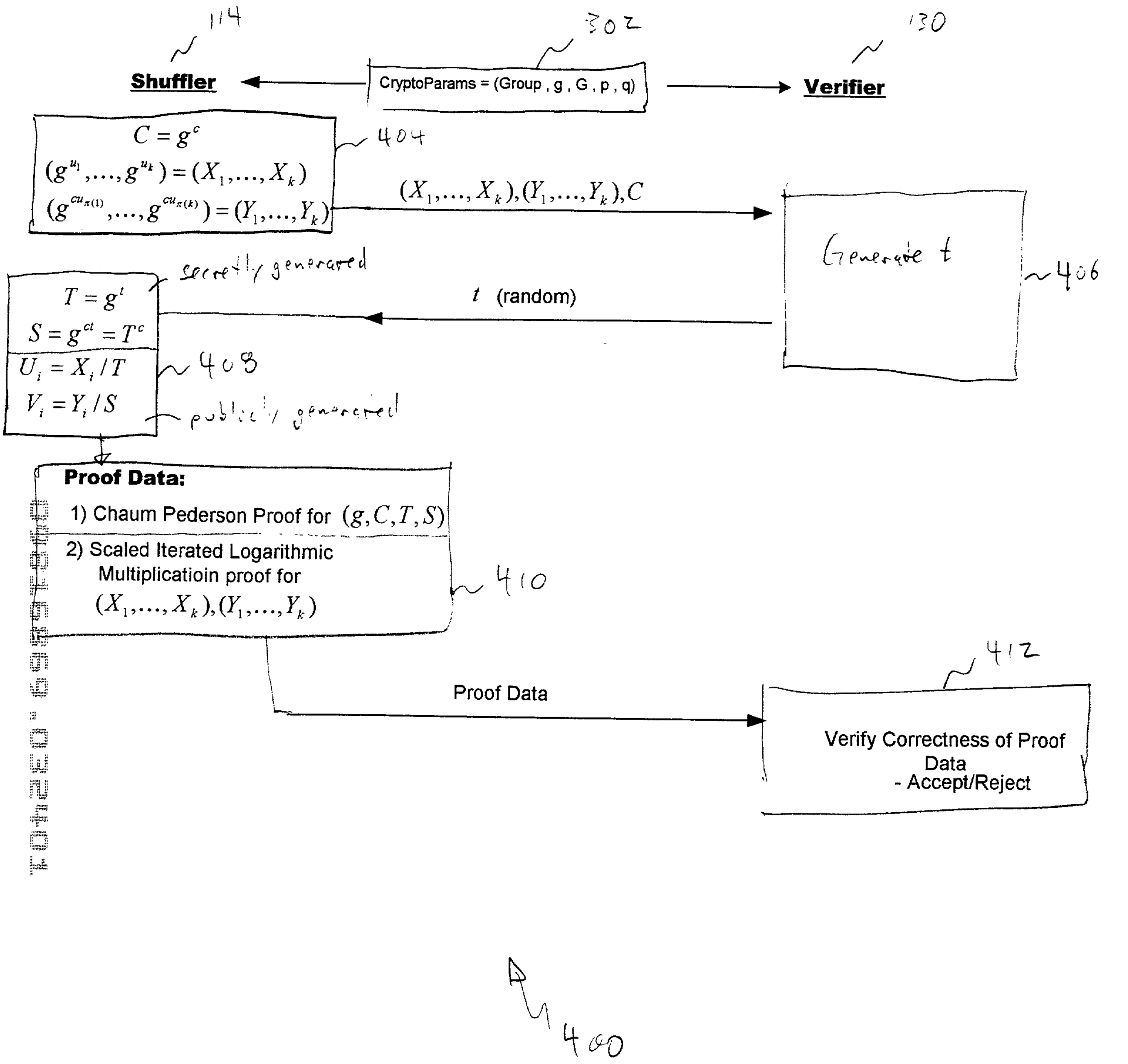

114
302
130
Shuffler
CryptoParams = (Group , g , G , p , q)
Verifier
$C = g^c$
$(g^{u_1},\ldots,g^{u_k}) = (X_1,\ldots,X_k)$
$(g^{cu_{\pi(1)}},\ldots,g^{cu_{\pi(k)}}) = (Y_1,\ldots,Y_k)$
404
$(X_1,\ldots,X_k),(Y_1,\ldots,Y_k),C$
Generate t
406
secretly generated
$T = g^t$
$S = g^{ct} = T^c$
t (random)
$U_i = X_i / T$
$V_i = Y_i / S$
408
publicly generated
Proof Data:
1) Chaum Pederson Proof for (g,C,T,S)
2) Scaled Iterated Logarithmic Multiplicatioin proof for $(X_1,\ldots,X_k),(Y_1,\ldots,Y_k)$
410
412
Proof Data
Verify Correctness of Proof Data
- Accept/Reject
400

Fig 4

General Shuffle
(Shuffler does not know exponents)

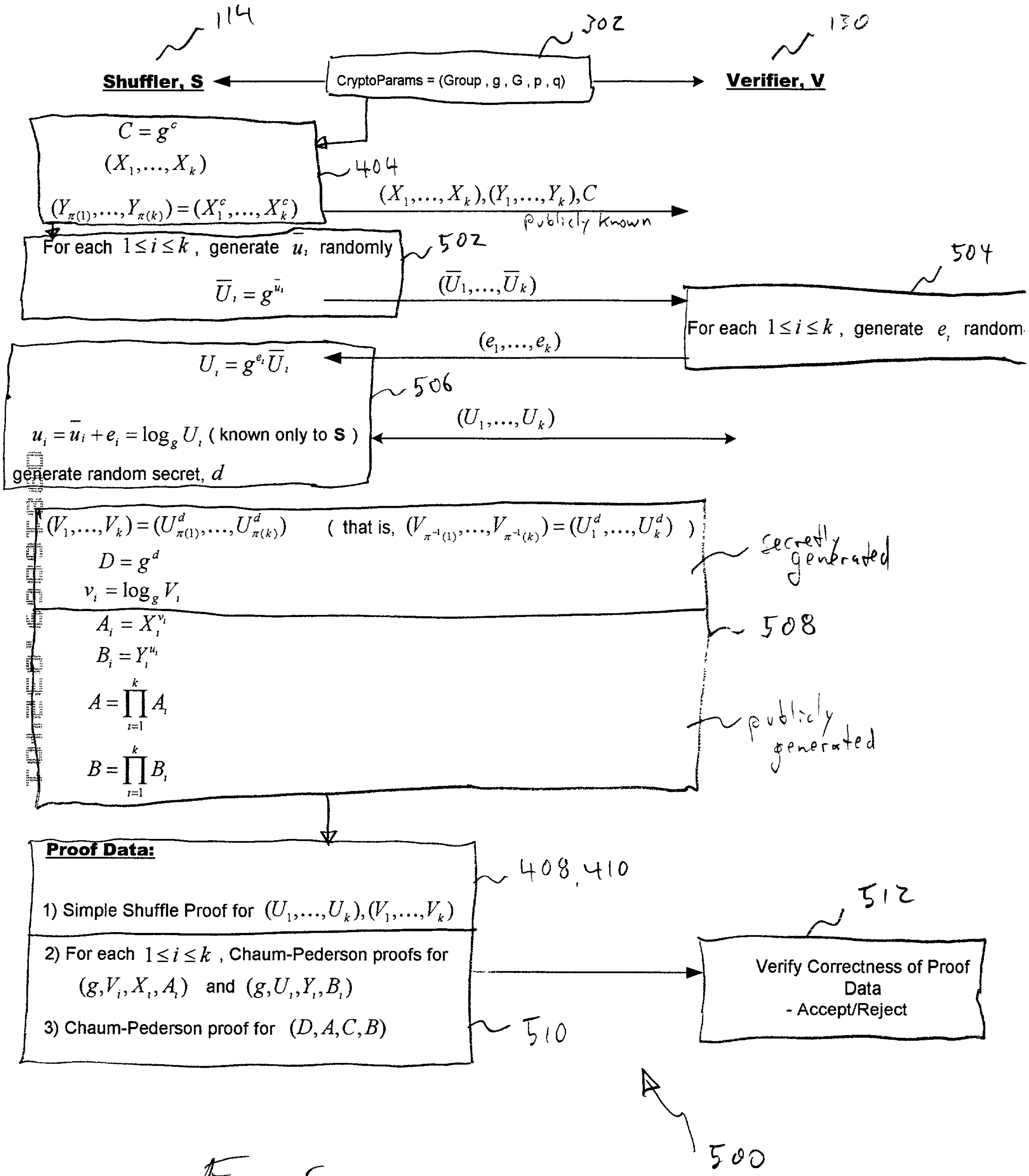
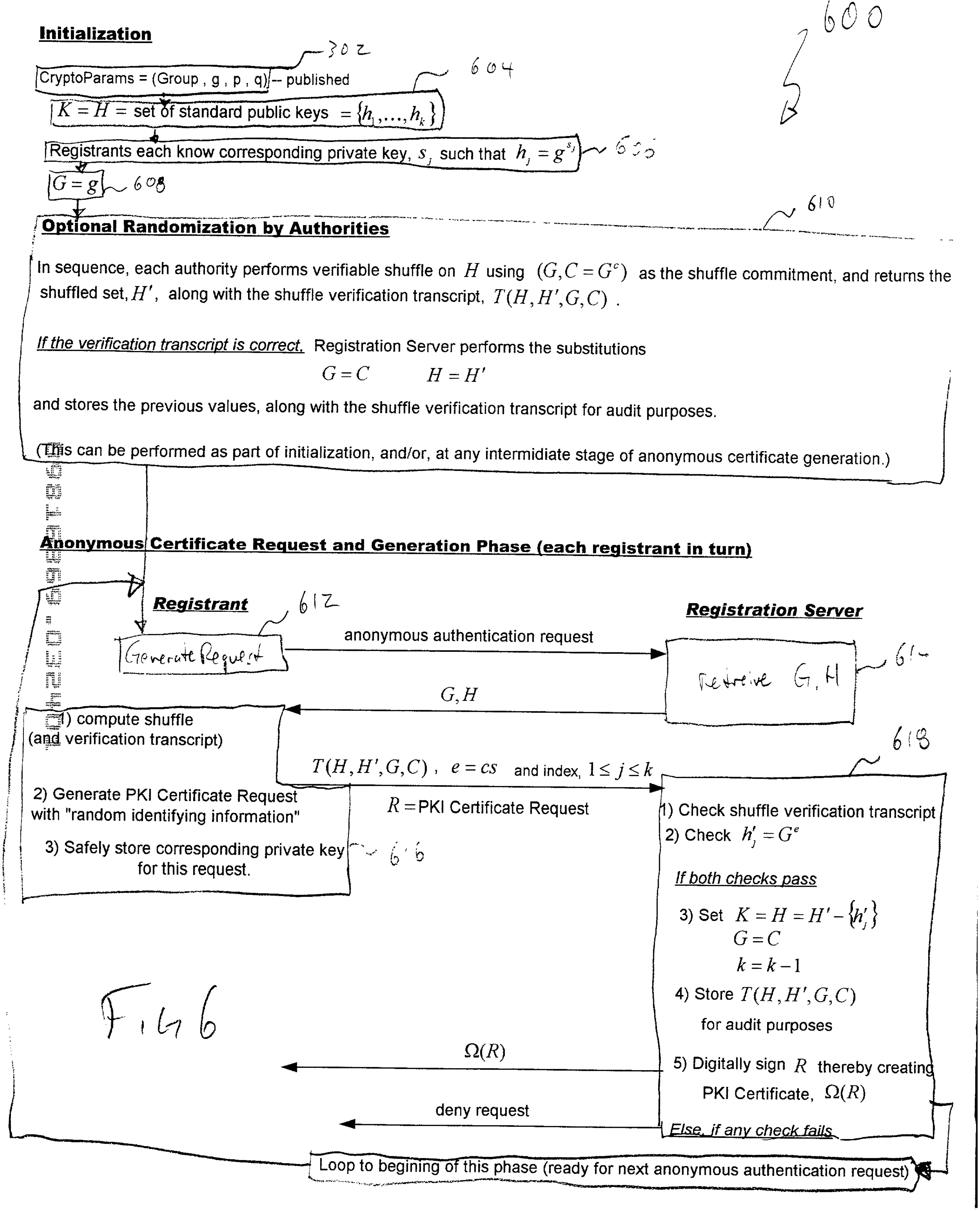

114
302
130
Shuffler, S
CryptoParams = (Group , g , G , p , q)
Verifier, V
$C = g^c$
$(X_1, \ldots, X_k)$
$(Y_{\pi(1)}, \ldots, Y_{\pi(k)}) = (X_1^c, \ldots, X_k^c)$
404
$(X_1, \ldots, X_k), (Y_1, \ldots, Y_k), C$
publicly known
For each $1 \le i \le k$, generate $\bar{u}_i$ randomly
$\bar{U}_i = g^{\bar{u}_i}$
502
$(\bar{U}_1, \ldots, \bar{U}_k)$
504
For each $1 \le i \le k$, generate e_i random
$(e_1, \ldots, e_k)$
$U_i = g^{e_i} \bar{U}_i$
506
$u_i = \bar{u}_i + e_i = \log_g U_i$ (known only to S)
$(U_1, \ldots, U_k)$
generate random secret, d
$(V_1, \ldots, V_k) = (U_{\pi(1)}^d, \ldots, U_{\pi(k)}^d)$ (that is, $(V_{\pi^{-1}(1)}, \ldots, V_{\pi^{-1}(k)}) = (U_1^d, \ldots, U_k^d)$)
$D = g^d$
$v_i = \log_g V_i$
Secretly generated
$A_i = X_i^{v_i}$
$B_i = Y_i^{u_i}$
$A = \prod_{i=1}^{k} A_i$
$B = \prod_{i=1}^{k} B_i$
508
publicly generated
Proof Data:
1) Simple Shuffle Proof for $(U_1, \ldots, U_k), (V_1, \ldots, V_k)$
408, 410
2) For each $1 \le i \le k$, Chaum-Pederson proofs for (g, V_i, X_i, A_i) and (g, U_i, Y_i, B_i)
3) Chaum-Pederson proof for (D, A, C, B)
510
Verify Correctness of Proof Data - Accept/Reject
512
500

Fig 5

Anonymous Certificate Distribution
(Protocol Variant 1)



Initialization

CryptoParams = (Group , g , p , q) -- published

$K = H =$ set of standard public keys $= \{h_1, \ldots, h_k\}$

Registrants each know corresponding private key, s_j such that $h_j = g^{s_j}$

$G = g$

Optional Randomization by Authorities

In sequence, each authority performs verifiable shuffle on H using $(G, C = G^c)$ as the shuffle commitment, and returns the shuffled set, H', along with the shuffle verification transcript, $T(H, H', G, C)$.

If the verification transcript is correct. Registration Server performs the substitutions

$$G = C \qquad H = H'$$

and stores the previous values, along with the shuffle verification transcript for audit purposes.

(This can be performed as part of initialization, and/or, at any intermidiate stage of anonymous certificate generation.)

Anonymous Certificate Request and Generation Phase (each registrant in turn)

Registrant

Generate Request

Registration Server

anonymous authentication request

Retreive G, H

G, H

1) compute shuffle (and verification transcript)

2) Generate PKI Certificate Request with "random identifying information"

3) Safely store corresponding private key for this request.

$T(H, H', G, C)$, $e = cs$ and index, $1 \leq j \leq k$

$R =$ PKI Certificate Request

1) Check shuffle verification transcript

2) Check $h'_j = G^e$

If both checks pass

3) Set $K = H = H' - \{h'_j\}$
$G = C$
$k = k - 1$

4) Store $T(H, H', G, C)$ for audit purposes

5) Digitally sign R thereby creating PKI Certificate, $\Omega(R)$

$\Omega(R)$

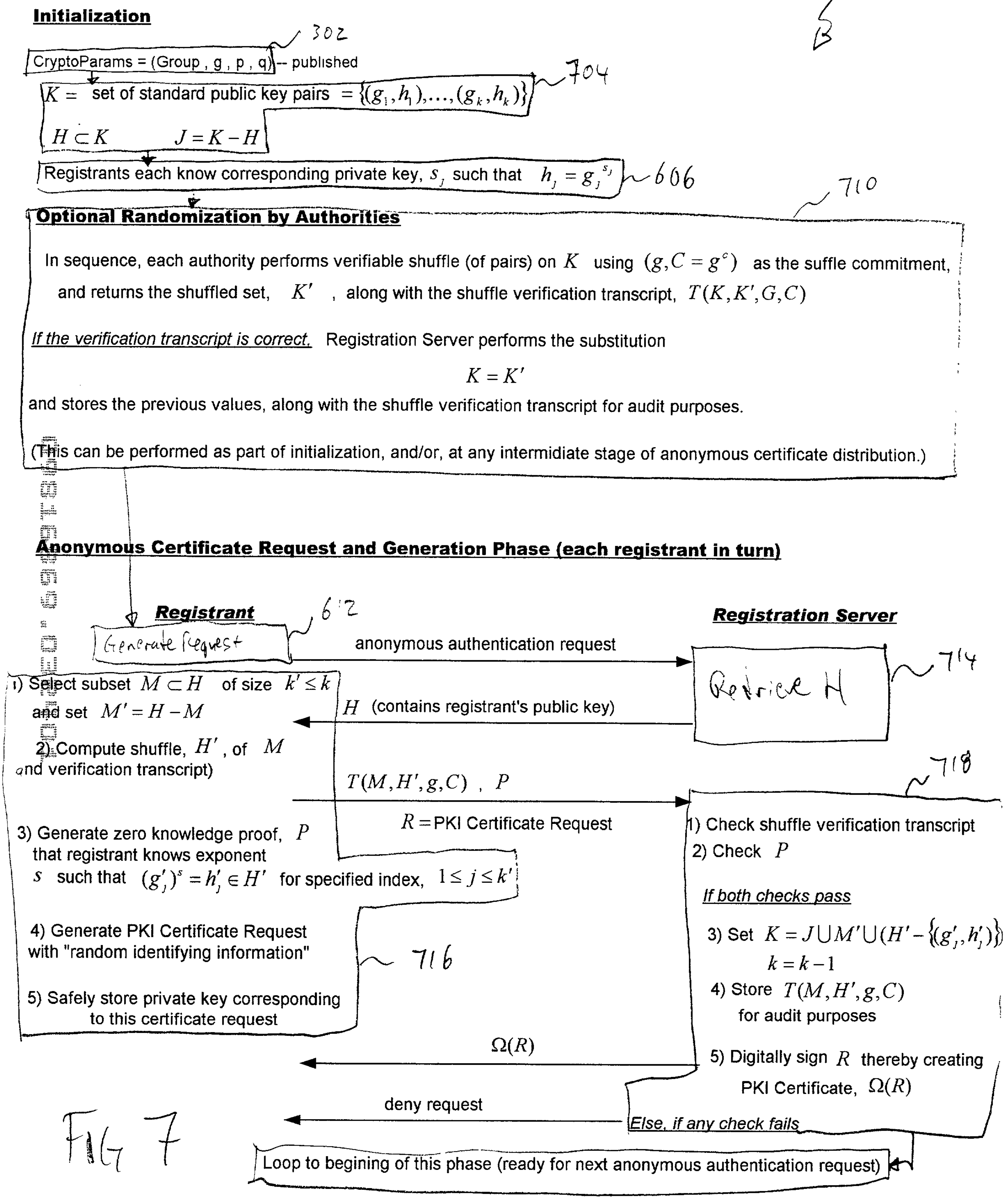
deny request

Else, if any check fails

Loop to begining of this phase (ready for next anonymous authentication request)

Fig 6

Anonymous Certificate Distribution (Protocol Variant 2)



Initialization

CryptoParams = (Group , g , p , q) -- published

$K =$ set of standard public key pairs $= \{(g_1,h_1),\ldots,(g_k,h_k)\}$

$H \subset K \qquad J = K - H$

Registrants each know corresponding private key, s_j such that $h_j = g_j^{s_j}$

Optional Randomization by Authorities

In sequence, each authority performs verifiable shuffle (of pairs) on K using $(g, C = g^c)$ as the suffle commitment, and returns the shuffled set, K' , along with the shuffle verification transcript, $T(K,K',G,C)$

If the verification transcript is correct. Registration Server performs the substitution

$$K = K'$$

and stores the previous values, along with the shuffle verification transcript for audit purposes.

(This can be performed as part of initialization, and/or, at any intermidiate stage of anonymous certificate distribution.)

Anonymous Certificate Request and Generation Phase (each registrant in turn)

Registrant

Registration Server

Generate Request

anonymous authentication request

Retrieve H

H (contains registrant's public key)

1) Select subset $M \subset H$ of size $k' \leq k$ and set $M' = H - M$

2) Compute shuffle, H' , of M and verification transcript)

$T(M,H',g,C)$, P

$R =$ PKI Certificate Request

1) Check shuffle verification transcript

2) Check P

3) Generate zero knowledge proof, P that registrant knows exponent s such that $(g'_j)^s = h'_j \in H'$ for specified index, $1 \leq j \leq k'$

4) Generate PKI Certificate Request with "random identifying information"

5) Safely store private key corresponding to this certificate request

If both checks pass

3) Set $K = J \cup M' \cup (H' - \{(g'_j, h'_j)\})$
$k = k - 1$

4) Store $T(M,H',g,C)$ for audit purposes

5) Digitally sign R thereby creating PKI Certificate, $\Omega(R)$

$\Omega(R)$

deny request

Else, if any check fails

Loop to begining of this phase (ready for next anonymous authentication request)

FIG 7